Übungen zur Vorlesung
Betriebssysteme II
Sommersemester 2006

AG BS
Jan Peleska

Blatt 3

Ein stark fairer Scheduler

Wir definieren einen neuen Scheduler *SCHED2*, der sich an der Konzeption des in der Vorlesung vorgestellten universell stark fairen Schedulers orientiert, aber auf folgende Weise implementierbar wird:

- *SCHED2* verwendet die `runqueue`-Struktur des Linux O(1) Schedulers; es wird aber nur *ein* `prio_array` benutzt – also kein Wechsel zwischen "active" und "expired".
- Auf die Universalität wird (natürlich) verzichtet.
- *SCHED2* lässt unbeschränkt viele Prozesse zu.
- Wenn ein rechnender Prozess die CPU nicht freiwillig abgibt, kann er sie bis zum Ablauf einer konstanten Zeitscheibe behalten. Danach wird zwangsweise ein anderer Prozess selektiert, wenn vorhanden.
- *SCHED2* vergibt Prozessgewichte (= Prioritäten) z_i im Bereich $z_i \in \{\ 0..MAX\_PRIO-1\ \}$. Wie beim O(1) Scheduler ist 0 die beste Priorität.
- Initial und nach Inanspruchnahme der CPU erhält jeder Prozess eine zufällige Priorität im Bereich $z_i \in \{\ 70..MAX\_PRIO-1\ \}$.
- Wenn im `prio_array` kein Prozess vermerkt ist, der Priorität 0 hat, werden die Prioritäten z_i der rechenbereiten Prozesse, welche nicht selektiert wurden, um 1 dekrementiert. Dies führt zu einem entsprechenden dequeue/enqueue im `prio_array`, um den Prozess gemäß seiner neuen (besseren) Priorität einzuordnen.
- Wenn im `prio_array` mindestens ein Prozess vermerkt ist, der Priorität 0 hat (also liste `queue[0]` nicht leer), werden alle Prozesse mit Priorität 0 in der durch `queue[0]` vorgegebenen Reihenfolge abgearbeitet. Während dieser Phase werden die Prioritäten rechenbereiter Prozesse *nicht* dekrementiert.

Aufgabe 1: Implementierung des fairen Schedulers (40%)

Wechseln Sie die erforderlichen Teile des existierenden Linux O(1) Schedulers – unter Beibehaltung der Schnittstellen und Datenstrukturen für die Runqueue etc. – gegen Ihre Implementierung des neuen Schedulers *SCHED2* aus.

Aufgabe 2: Test des fairen Schedulers (20%)

Testen Sie die Wirkung von *SCHED2* mit Hilfe Ihrer Anwendung zur Ressourcenverwaltung aus Blatt1. Ändert sich bei *SCHED2* im Vergleich zum Linux-O(1) Scheduler die Laufzeit, bis alle Ressourcen verbraucht sind ?

Aufgabe 3: Fairnessbeweis (40%)

Geben Sie eine modifizierte Version des in der Vorlesung vorgetragenen Beweises der starken Fairness, angepasst auf die Scheduling Strategie des neuen Schedulers *SCHED2*. Warum ist *SCHED2* nicht mehr von der Komplexitätsklasse O(1)? Welche Komplexitätsklasse hat er?

Abgabe: Bis Mittwoch, 07.06.2006, in der Übung.

Sowohl bei der schriftlichen Lösung als auch im Source-Code die Namen aller Gruppenmitglieder nicht vergessen!